# Better Tower Defense – Design Document

Jacob Dufault

## System Architecture – 5000 miles

Unity is the Unity engine; the Entity system is described below; Networking relates to interfacing with other computes; the 3rd party code involves many systems, such as tweening, GUI work, serialization (protobuf), potentially inversion-of-control/dependency injection…. Systems and Data are all of the gameplay logic.

Hopefully it is obvious, but this is an extremely simplified model of the engine; the actual models would be much too long – the Entity system model below does not even go into full detail, yet it has surprisingly complexity even in the simplified form.

Unity

Entity System

Networking

3rd party: Tweening, GUI, Serialization, IoC/DI, …

## System Architecture Overview - Entities

An overview of the entity system, with relevant modules grouped together. The EntityManager coordinates all of the work; the Entity contains Data; the DataAccessor is used for quick retrieval of Data; the EntityCommand is structured user input, whereas the EntityCommandNetworkManager issues EntityCommands to the EntityManager; the SystemBehavior is user-defined logic that operates on Entities, which is selects by using a combination of an Invoke and a set of Filters (which are processed to form a CompiledFilter); DataProviders mediates between the Unity entity system and this one. GameData<T> and GameDataProvider<T> provide type erasure for Data and DataProvider so that they can be used generically through the EntitySystem but client code has type safety.

EntityManager

Data

GameData<T>

DataAccessor

Entity

EntityCommand

EntityCommandNetworkManager

CompiledFilter

Invoke

SystemBehavior

DataProvider

GameDataProvider<T>>

# Feature Implementation Notes

## Entity System

See the architecture overview for the Entity System.

## Networking

Connect to a server with the given password

*Connect(string ip, string password)*

Disconnect from a server or shut the server down

*Disconnect()*

Start running a server with the given password

*StartServer(string password)*

Send a network command, ie, pause the game.

*SendCommand(NetworkCommand command)*

Listen to a network message, ie, when we receive a pause game message.

*AddCommandListener(Type commandType, Action<NetworkCommand> listener)*

## Unit Spawning

Uses the entity system; listen for objects which have “SpawnWave” data and spawn based on that data. No API.

## Locomotion

Uses the entity system; listen for objects which have “PathingData” and move based on that (ie, it will contain a target, a path to follow, …). No API.

## Effects

Uses the entity system; listen for objects which have “Effects” attached to them and change the object status based on said effects. No API.

## Building Placement

Attempt to build a new building at the given location; return if successful. Called via the network messaging system; UI creates network messages and sends them.

*bool BuildAt(vec3 location, GameObject prefab)*

## Resources

Uses the entity system; listen for buildings which can modify resource state and update based on them. No API.

## Power System

Use the effect system; tie into spatial optimization systems; no API.

## Dynamic Difficulty

Use the entity system; modify spawning data (so it ties into the spawning system). Uses a separate statistics module which just monitors the game gathering relevant statistics (number of enemies killed, number of players, average enemy life span, …). No API.